Juliana Hildebrandt, Till Kolditz, Dirk Habich, Wolfgang Lehner

Lower Bound-oriented Parameter Calculation for AN Coding

Erstveröffentlichung in / First published in:

International Symposium on Information Theory and Its Applications, ISITA. Singapore, 28.-31.10.2018. IEEE, S. 590-594. ISBN 978-4-88552-318-2.

DOI: http://dx.doi.org/10.23919/ISITA.2018.8664399

Final edited form was published in " 2018 International Symposium on Information Theory and Its Applications (ISITA). Singapore 2018", S. 590-594, ISBN 978-4-88552-318-2
https://doi.org/10.23919/ISITA.2018.8664399

Lower Bound-oriented Parameter Calculation for AN Coding

Juliana Hildebrandt Till Kolditz Dirk Habich Wolfgang Lehner
Database Systems Group
Technische Universität Dresden, Nöthnitzer Straße 46, 01187 Dresden
{juliana.hildebrandt, till.kolditz, dirk.habich, wolfgang.lehner}@tu-dresden.de

***Abstract*—The hardware as well as software communities have recently experienced a shift towards mitigating bit flips issues in software, rather than completely mitigating only in hardware. For this software error mitigation, arithmetic error coding schemes like AN coding are increasingly applied because arithmetic operations can be directly executed without decoding and bit flip detection is provided in an end-to-end fashion. In this case, each encoded data word is computed by multiplying the original data word with a constant integer value A. To reliably detect b bit flips in each code word, the value A has to be well-chosen, so that a minimum Hamming distance of $b+1$ can be guaranteed. However, the value A depends on the data word length as well as on the desired minimum Hamming distance. Up to now, a very expensive brute force approach for computation of the value for A is applied. To tackle that in a more efficient way, we present a lower bound-oriented approach for this calculation in this paper.**

I. Introduction

Recent studies have shown that future hardware is becoming less and less reliable and that multi-bit flips may prevail over single bit flips [1]–[4]. Scaling today's single bit flip-centric hardware-based protection techniques to cover multi-bit flips is possible, but this introduces large performance, chip area, and power overheads, which will become non-affordable in the future [1], [4]. Thus, a shift towards mitigating these reliability issues at higher software layers, rather than completely mitigating these issues only in hardware is observable [4]–[6]. For this software-based error mitigation, arithmetic error coding schemes (AN coding) are a well-known and effective technique [7], [8]. Most recently, AN coding is applied in different domains like compiler [9]–[11] or database systems [12]. Advantages of AN coding are: (1) arithmetic operations can be directly executed without decoding and (2) bit flip detection is provided in an end-to-end fashion [8]–[12]. The latter means that after encoding data, errors are detectable during data transportation, data storage, and data processing.

Problem Statement: AN coding is a very simple error detection code, because encoded data words—code words—are computed by multiplying each original information word with a constant integer value A [7], [8]. As a result of this multiplication, the domain of code words expands such that only multiples of A become valid code words and all others are considered as invalid code words. Now, to reliably detect b bit flips in each code word, a value for A has to be used which guarantees a minimum Hamming distance of $b+1$, whereby A depends on the information word length l and on the number of detectable bit flips b [8], [12]. Moreover, it is usually not some arbitrary value for A sought but a small one, so that the domain of code words is small [8], [12]. Unfortunately, only a very expensive brute force approach for calculation of this parameter value for A exists. Therefore, only few As are known covering few combinations of l and b [8], [12].

Our Contribution and Outline: To overcome that, we developed a novel computation technique for the parameter value A based on equivalence classes of code word pairs. In detail, we make the following contributions in this paper:

1) We briefly recap necessary preliminaries in Section II. In particular, we review the brute force approach for the calculation of the parameter value A.
2) Then, we introduce our novel approach to compute a lower bound on the minimum Hamming distance of a code based on code word differences in Section III.
3) In Section IV, we apply our lower bound approach to compute the parameter value A in a more efficient way.
4) In Section V, we present our evaluation results.

Finally, we conclude the paper with a summary in Section VI.

II. Preliminaries

AN coding is an error detecting code preserving arithmetic operations [7], [8]. Let $\mathbb{D} = [0, 2^l)$ with $l \in \mathbb{N}$ be a set of information words, then the code words c are computed by multiplying each original information word $d \in \mathbb{D}$ with a constant integer value A, $c = d \cdot A$. We denote the set of all code words as $\mathbb{C}^A_{[0,2^l)}$. As a result of this encoding, all multiples m of A, $0 \leq m < A \cdot 2^l$ are valid code words, but all other values—including multiples $m', m' < 0 \vee m' \geq A \cdot 2^l$—are non-valid code words. Let k be the bit width of A ($k = \lceil \log_2 A \rceil$). Thus, the greatest code word $A \cdot (2^l - 1)$ can be encoded binarily with $k + l$ bits, because

$$\log_2\lceil A \cdot (2^l - 1)\rceil \leq \log_2\lceil A\rceil + \log_2\lceil 2^l - 1\rceil = k + l. \quad (1)$$

We define the common code word length $k+l$ as n. Since the bits of the information word and the additional bits are non-separable in each code word, AN coding is a non-systematic error code. For an efficient decoding, an odd A should be used, so that a multiplication $d = A^{-1} \cdot c \mod 2^l$ with the multiplicative inverse $A^{-1} \in \mathbb{Z}_{2^l}$ can be applied instead of a division $d = c/A$ for decoding [12]. An error is detected using $c \not\equiv 0 \mod A$ [8]–[12].

AN coding has only one parameter A which has a high impact on the error detection capability [7], [8], [12]. Based on that, the most challenging issue for AN coding is to find an appropriate value for A for given l and b with $l \in \mathbb{N}$ as the information word length and $b \in \mathbb{N}$ as the number of bit errors that should be reliably detected. Additionally, the value for A should be as small as possible (length k), so that the resulting code word length $n = l + k$ is as small as possible as well [8], [12]. Up to now, a brute force approach consisting of two components is applied for this parameter calculation.

Component 1 determines the minimum Hamming distance for given A and l and we denote this *minimum Hamming distance* as $d_{\min}(\mathbb{C}^A_{[0,2^l)})$. This is the smallest number of bits, which has to be changed to convert one code word into another one. Consequently, the number of bit flips which can always be reliably detected is $d_{\min}(\mathbb{C}^A_{[0,2^l)}) - 1$. To calculate this minimum Hamming distance $d_{\min}(\mathbb{C}^A_{[0,2^l)})$, the brute force approach iterates over all pairs of distinct code words $(c, c') \in (\mathbb{C}^A_{[0,2^l)})^2$ and determines the Hamming distances $d_H(c, c')$ for each pair. Then, the minimum of these distances is $d_{\min}(\mathbb{C}^A_{[0,2^l)})$ for given A and l.

Component 2 computes a small value for A for given l and b by looping over all possible odd As starting with $A = 3$. For each A, the minimum Hamming distance $d_{\min}(\mathbb{C}^A_{[0,2^l)})$ is computed using *Component 1* [12]. This iteration ends with the first A satisfying the condition $d_{\min}(\mathbb{C}^A_{[0,2^l)}) = b + 1$.

III. Lower Bound for Minimum Hamming Distance Based on Code Word Differences

To avoid the brute force computation, we will now introduce our novel approach in this section. For that, we abstract from AN coding as a specific error code by looking at a common error code $\mathbb{C}$ with a given set of code words in the domain of $[0, 2^n)$. Based on that, we are able to interpret code words as natural numbers and can compute differences between two arbitrary code words. With the help of this difference consideration, we can determine a lower bound for the minimum Hamming distance of this code $\mathbb{C}$ in a non-brute force way.

A. Delta equivalence classes

Let n be a natural number (code word length) and let $m, m' \in [0, 2^n), m \leq m'$ be natural numbers (code words). Then, for each difference $\Delta \in \mathbb{N}, \Delta < 2^n$ exist $2^n - \Delta$ pairs of values such that $m' - m = \Delta$. Thus, we create equivalence classes for pairs of values in $[0, 2^n)$ with the same difference Δ:

$$[\Delta] = \{(m, m') | m, m' \in [0, 2^n), \Delta = m' - m, m \leq m'\}. \tag{2}$$

Example 1. Let $n = 4$. Thus

$$\begin{aligned}&[2] = \{(0,2), (1,3), (2,4), ..., (13,15)\} \text{ and}\\ &[3] = \{(0,3), (1,4), (2,5), ..., (12,15)\}\end{aligned} \tag{3}$$

are equivalence classes for the differences 2 and 3.

B. Signed-Digit Expansions for Differences

Let $m, 0 \leq m < 2^n$ be a natural number and $\underline{m}_p \in \{-1, 0, 1\}$, $0 \leq p < n$ with the condition

$$m = \sum_{p=0}^{n-1} \underline{m}_p \cdot 2^p. \tag{4}$$

We call $\underline{m}_{n-1} \dots \underline{m}_1 \underline{m}_0$ a *signed-digit expansion* (SDE) with length n of a number m. This representation is redundant [13], [14], because each number m can be represented by different SDEs. The *(Hamming) weight of an SDE* $\underline{m}_{n-1} \dots \underline{m}_1 \underline{m}_0$ is defined in [14] as the number of its non-zero digits. In the following, the symbol $\underline{1}$ is used for -1.

Example 2. Let $n = 4$ and $m = 3$. Then, $\mathtt{0011}$, $\mathtt{010\underline{1}}$, $\mathtt{01\underline{1}1}$, $\mathtt{1\underline{1}0\underline{1}}$, and $\mathtt{1\underline{1}\underline{1}1}$ are all SDEs for m with length 4, where the weights are 2, 2, 3, 3, and 4[1].

For each equivalence class $[\Delta]$, $\Delta \in \mathbb{N}$, we can divide the pairs $m, m' \in \mathbb{N}$ in further equivalence classes of SDEs with length n for Δ:

$$\begin{aligned}&[\underline{\Delta}_{n-1} \underline{\Delta}_1 \underline{\Delta}_0]\\ &= \{(m, m') \mid (m, m') \in [\Delta], \forall p : m'_p - m_p = \underline{\Delta}_p\}\end{aligned} \tag{5}$$

where m_p and m'_p are the values $\in \{\mathtt{0}, \mathtt{1}\}$ at position p in the binary representation of m and m'. Thus, an equivalence class with the digit $\mathtt{1}$ at position p contains only pairs (m, m') fulfilling the condition that m' has digit $\mathtt{1}$ at position p and m has digit $\mathtt{0}$ at position p. Similarly, an equivalence class with the digit $\underline{\mathtt{1}}$ at position p contains only pairs (m, m') fulfilling the condition that m' has digit $\mathtt{0}$ at position p and m has digit $\mathtt{1}$ at position p. Thus, an equivalence class with a non-zero digit at position p contains only pairs (m, m') with different digits at position p. Consequently, the weight of an SDE is the Hamming distance of all pairs in the corresponding equivalence class.

Example 3. Let $n = 4$ and $\Delta = 3$. Thus

$$\begin{aligned}
&[\mathtt{0011}] = \{(0,3), (4,7), \dots\} &&\Rightarrow d_H(0,3) = d_H(4,7) &&= 2\\
&[\mathtt{010\underline{1}}] = \{(1,4), (3,6), \dots\} &&\Rightarrow d_H(1,4) = d_H(3,6) &&= 2\\
&[\mathtt{01\underline{1}1}] = \{(2,5), (10,13)\} &&\Rightarrow d_H(2,5) = d_H(10,13) &&= 3\\
&[\mathtt{1\underline{1}0\underline{1}}] = \{(5,8), (7,10)\} &&\Rightarrow d_H(5,8) = d_H(7,10) &&= 3\\
&[\mathtt{1\underline{1}\underline{1}1}] = \{(6,9)\} &&\Rightarrow d_H(6,9) = 4
\end{aligned} \tag{6}$$

To summarize, for each difference Δ of natural numbers m, m', there is at least one delta equivalence class with a minimal number of non-zero digits. This minimal number is a first lower bound on the Hamming distance of all pairs $(m, m'), \Delta = m' - m$.

C. Minimal Hamming Distance of Pairs with Difference Δ

A *non-adjacent form (NAF)* is an SDE without adjacent non-zero digits. As it is known from [13], for a given number Δ, the minimal weight of an SDE of Δ is the weight of the NAF. Additionally, each number Δ has a unique NAF (with an arbitrary number of leading zeros).

[1]For fixed n, there is only a finite number of SDEs for each $m, 0 \leq m < 2^n$. For an arbitrary n, there is an infinite number of SDEs for each integer $m \neq 0$ [13].

Final edited form was published in " 2018 International Symposium on Information Theory and Its Applications (ISITA). Singapore 2018", S. 590-594, ISBN 978-4-88552-318-2
https://doi.org/10.23919/ISITA.2018.8664399

Example 4. The SDEs $\mathtt{11}$, $\mathtt{1\underline{1}1}$, $\mathtt{1\underline{1}0\underline{1}}$, $\mathtt{1\underline{1}\underline{1}1}$, $\mathtt{1\underline{1}\underline{1}0\underline{1}}$, $\mathtt{1\underline{1}\underline{1}\underline{1}1}$ etc. are adjacent forms of the value 3. The non-adjacent form of 3 is $\mathtt{10\underline{1}}$.

From [15] we know, that the weight of the NAF of a natural number Δ is the number of $\mathtt{1}$-digits in the result of the binary bit operation $XOR(3\Delta, \Delta)$. We also know from [15], that there is a sequence $a(\Delta)$ calculating the weight of the NAF for each $\Delta \in \mathbb{N}$. This is OEIS sequence $A007302$:

$$a(0) = 0, a(\Delta) = \begin{cases} a(\frac{\Delta}{2}) & \text{if } 2|\Delta, \\ 1 + a(\frac{\Delta\pm 1}{4}) & \text{if } 4|(\Delta \pm 1). \end{cases} \tag{7}$$

Both calculation approaches lead to the same result. Then, an open challenge is, that we have a given code length n and that the NAF of a number $\Delta > \frac{3}{4} \cdot 2^n$ has non-zero values at position $p = n$. This is a contradiction.

Example 5. Let $n = 7$ and $0 \le m < m' < 2^7$. We want to find the lower bound for Hamming distances of pairs of natural numbers (m, m') with the difference Δ. For differences from 1 to $\frac{3}{4} \cdot 2^7 = 96$ it is valid to use sequence $a(\Delta)$. But let $\Delta = 123$. The NAF of 123 is $\mathtt{10000\underline{1}0\underline{1}}$. This means, that all pairs $(m, m') \in [\mathtt{10000\underline{1}0\underline{1}}]$ require distance 3 and must have the following property. Like explained in Section III-B, the digit at position 7 of m' is $\mathtt{1}$. Thus, for the higher value m' must hold $m' > 2^7$, which is a contradiction. The equivalence class $[\mathtt{10000\underline{1}0\underline{1}}]$ contains no values[2].

For those cases, we have to find the SDE with length n with the smallest number of non-zero digits. All SDEs for natural numbers $\Delta > \frac{3}{4} \cdot 2^n$ start with at least two $\mathtt{1}$-digits. Those SDEs for $2^n - 2^x \le \Delta < 2^n$ with length n,a minimal x, and a minimum number of non-zero values start with $n-x$ $\mathtt{1}$-digits end with the NAF for $\Delta - 2^n + 2^x$. The value for x is

$$x = \lceil \log_2(2^n - \Delta) \rceil. \tag{8}$$

Example 6. Let $n = 7$ and $\Delta = 123$. The NAF of Δ is $\mathtt{10000\underline{1}0\underline{1}}$. It has length $n + 1 = 8$ and a non-zero value at position n. The SDE with length n and with the least number of non-zero values is $\mathtt{1111\ 10\underline{1}}$. First we have

$$x = \lceil \log_2(2^n - \Delta) \rceil = \lceil \log_2(5) \rceil = 3. \tag{9}$$

The SDE starts with $n - x = 7 - 3 = 4$ $\mathtt{1}$-digits. Second, the SDE ends with the NAF of

$$\Delta - 2^n + 2^x = 123 - 128 + 8 = 3, \tag{10}$$

which is $10\underline{1}$.

Thus, for $\Delta > \frac{3}{4} \cdot 2^n$, the number of non-zero digits in the SDE with length n and the least number of non-zero values is the sum of $n - x$ and $a(\Delta - 2^n + 2^x)$. This leads to the adapted (finite) sequence

$$\tilde{a}(\Delta) = \begin{cases} a(\Delta) & \text{if } \Delta \le \frac{3}{4} \cdot 2^n, \\ n - x + a(\Delta - 2^n + 2^x) & \text{else}, \end{cases} \tag{11}$$

[2]The reader will remark that there are values for Δ between $\frac{1}{2} \cdot 2^n$ and $\frac{3}{4} \cdot 2^n$ whose NAF has also more than n digits. Those NAFs have the form $\mathtt{10\underline{1}xxx...}$. The NAF of 86 is $\mathtt{10\underline{1}0\underline{1}0\underline{1}0}$. For those cases it can be shown that there exists an SDE of the form with length n and with the same minimal number of non-zero digits with the form $\mathtt{11xxx...}$. For 86 we have the SDE $\mathtt{110\underline{1}0\underline{1}0}$. Consequently, it is sufficient to consider only those values for Δ with $\Delta > \frac{3}{4} \cdot 2^n$.

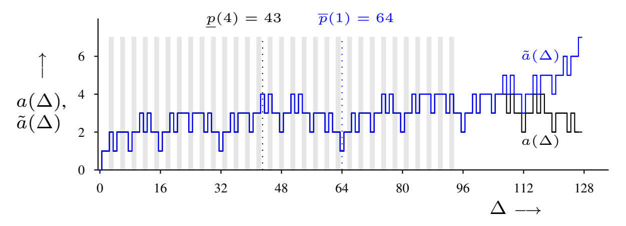


Fig. 1. Sequences $a(\Delta)$, and $\tilde{a}(\Delta)$ for $n = 7$, and multiples of $A = 3$

which is defined for $0 \le \Delta < 2^n$. Fig. 1 shows the comparison of the sequences $a(\Delta)$ and $\tilde{a}(\Delta)$. The determination with the XOR operation can be adapted for Δ by the sum of $n-x$ and the number of $\mathtt{1}$-digits in $XOR(3 \cdot (\Delta - 2^n + 2^x), \Delta - 2^n + 2^x)$. To summarize, every code $\mathbb{C}$ with a given set of code words in the domain of $[0, 2^n)$ (code length equals n) has a finite number of different code word differences. For each code word difference, there exist different SDE equivalence classes containing at least one pair of natural numbers. It might happen, that an SDE does not contain any elements. However, it can be guaranteed, that the minimum Hamming distance $d_{\min}$ of a code $\mathbb{C}$ is not smaller than the minimum of the SDE of the lowest weight of all code word differences Δ. But it can not be guaranteed, that an SDE equivalence class with length n and the least number of non-zero values contains at least one pair of code words. In those cases, another SDE equivalence class with a greater number of non-zero digits contains at least one pair of code words. Thus, the minimum Hamming distance $d_{\min}$ of a code is greater than or equal to the minimum of the weight of an SDE with length n of each possible code word difference (lower bound).

D. Optimizations

Two optimizations for two different situations are possible. First, let a code $\mathbb{C}$ and thus, its length n and the (ordered) set $\mathcal{D}$ of code word differences be given. To calculate the lower bound on $d_{\min}(\mathbb{C})$, we have to iterate over the code word differences $\Delta \in \mathcal{D}$ with the objective of finding the minimum $d_{\min} = \min\{\tilde{a}(\Delta) | \Delta \in \mathcal{D}\}$. We iterate over the code word differences in ascending order starting with the lowest code word difference. Initially, $d_{\min}$ can be set to the highest possible value, namely n. If for a difference Δ a smaller value $\tilde{a}(\Delta)$ is found, $d_{\min}$ is set to $\tilde{a}(\Delta)$. In the finite sequence $\tilde{a}(\Delta)$, there is one highest position $\overline{p}(d_{\min})$ where $d_{\min}$ occurs. All higher positions contain higher values. Thus, for each adapted intermediate result $d_{\min}$, we can calculate the highest position, where $d_{\min} - 1$ occurs. Greater code word differences Δ lead to higher values. The highest position, where a lower bound on the Hamming distance $d_{\min} - 1$ can occur, is

$$\overline{p}(d_{\min} - 1) = 2^n - 2^{n - d_{\min} + 1}. \tag{12}$$

Each time $d_{\min}$ is decreased during the iteration, the highest position to test can be set to $\overline{p}(d_{\min} - 1)$.

Example 7. Let $n = 7$ and a code with the differences $\{3, 6, 9, \ldots, 93\}$ be given. During the iteration, the actual value $d_{\min}$ is set to 2, because $\tilde{a}(3) = 2$. The last position Δ,

Final edited form was published in " 2018 International Symposium on Information Theory and Its Applications (ISITA). Singapore 2018", S. 590-594, ISBN 978-4-88552-318-2
https://doi.org/10.23919/ISITA.2018.8664399

where $\tilde{a}(\Delta) = d_{\min} - 1 = 1$ occurs, is $\overline{p}(d_{\min} - 1) = \overline{p}(1) = 2^7 - 2^6 = 64$. At this point we know, that we have to test all differences up to 64, because for all higher differences Δ holds $\tilde{a}(\Delta) \geq 2$ and the actual minimum of $d_{\min}$ is only decreased, if we find a difference with $\tilde{a}(\Delta) < 2$. Fig. 1 illustrates sequence $\tilde{a}(\Delta)$ for $n = 7$ and highlights the given code word distances as well as $\overline{p}(1) = 64$.

Second, let a code $\mathbb{C}$ be given. We want to decide, if the lower bound on $d_{\min}(\mathbb{C})$ is lower than a given value b. If $d_{\min}(\mathbb{C}) \geq b$, then for all code word distances $\Delta < \frac{3}{4} \cdot 2^n$ holds $a(\Delta) \geq b$. We know, that there is one lowest position $\underline{p}(a(\Delta))$ for each sequence element, where the weight $a(\Delta)$ occurs the first time. If the code contains pairs of code words with a lower difference than $\underline{p}(a(\Delta))$ (a difference at a lower sequence position), the code has a lower Hamming distance than b. This position can be derived from $a(\Delta)$ for $\Delta < \frac{3}{4} \cdot 2^n$. It holds $\underline{p}(0) = 0$, $\underline{p}(1) = 1$ and $\underline{p}(b) = 4 \cdot \underline{p}(b-1) - 1$. It can be shown by induction and is also known from [15, sequence A007583], that

$$\underline{p}(b) = \frac{2^{2b-1} + 1}{3}. \tag{13}$$

for $b > 0$. Thus, it holds for the smallest code word difference $\underline{\Delta}$ of a given code $\mathbb{C}$ and the given minimum Hamming distance b to test

$$\underline{\Delta} < \min\left(\frac{2^{2b-1} + 1}{3}, \frac{3}{4} \cdot 2^n\right) \Rightarrow d_{\min}(\mathbb{C}) < b. \tag{14}$$

Example 8. Let $n \geq 7$ and $b = 4$. The smallest possible code word difference for a code with the Hamming distance 4 is

$$\underline{p}(4) < \frac{2^{2\cdot 4-1} + 1}{3} = 43. \tag{15}$$

Thus, the smallest possible code word difference of a code with the Hamming distance $d_{\min}(\mathbb{C}) \geq 4$ is 43. For codes with $n \geq 7$ and a lower smallest code word difference holds $d_{\min}(\mathbb{C}) < 4$. Fig. 1 shows the value $\underline{p}(4)$.

IV. Lower Bound-oriented Parameter Calculation for AN Coding

As presented in Section II, to compute a small value A for given l (information word length) and b (guaranteed maximum number of detectable bit flips) two components are important. With the lower bound approach of the previous section, we can optimize this computation as follows.

A. *Optimization of Component* 1

To optimize the computation of the minimum Hamming distance for an AN code with given l and A, we can easily apply our introduced lower bound approach, because all code word distances are multiples m of A, $0 < m \leq A \cdot (2^l - 1)$.

Example 9. Let $A = 3$ with $(k = 2)$ and $l = 5$, thus $n = l + k = 7$. Fig. 1 shows the sequences $a(\Delta)$ and the adapted sequence $\tilde{a}(\Delta)$. Gray bars highlight the relevant multiples of 3, the values from which we have to determine the lower bound on the minimum Hamming distance for this setting.

```
 1: procedure CALCA($b$, $l$)
 2:     $A \leftarrow \frac{2^{2b-1}+1}{3}$                          ▷ Smallest possible $A$
 3:     found $\leftarrow \perp$
 4:     while !found do
 5:         $n \leftarrow \lceil \log_2(A) \rceil + l$            ▷ Sum of $l$ and bit width of $A$
 6:         $\overline{p} \leftarrow \min(A \cdot (2^l - 1), 2^n - 2^{n-b+1})$
 7:         found $\leftarrow \top$
 8:         $m \leftarrow A$                                   ▷ Odd multiple of $A$
 9:         while $m \leq \overline{p} \wedge$ found do          ▷ Iterating over the multiples
10:             if $\tilde{a}(m) < b$ then
11:                 found $\leftarrow \perp$
12:             end if
13:             $m \leftarrow m + 2A$                          ▷ Next odd multiple
14:         end while
15:         if !found then
16:             $A \leftarrow A + 2$                           ▷ Next odd $A$
17:         end if
18:     end while
19:     return $A$
20: end procedure
```

Fig. 2. Calculation of the smallest A guaranteeing the detection of $b - 1$ bit flips for a given information word length l.

To determine the minimum Hamming distance of an AN code with given A and l, we simply have to iterate over all multiples of A from A to $A \cdot (2^l - 1)$. Furthermore the first optimization explained in Section III-D can be applied. There is a further AN-code specific optimization. Because of $\tilde{a}(2\Delta) = \tilde{a}(\Delta)$ for $\Delta \leq \frac{3}{4} \cdot 2^n$ and $\tilde{a}(2\Delta) \geq \tilde{a}(\Delta)$ otherwise, we do not have to test even multiples of A.

B. *Optimization of Component* 2

In order to determine the smallest value for A for a given information word length l and a desired minimum Hamming distance of b (to detect up to $b - 1$ bit flips), we usually have to iterate over all odd values for $A \geq 3$ and to check the lower bound on the minimum Hamming distance of each code as explained in Section IV-A. Here, the second optimization described in Section III-D can be easily applied if $l > 1$. Thus, we only have to loop over odd values for A with $A \geq \underline{p}(b)$, because the smallest code word difference is A.

C. *Lower Bound-oriented Algorithm*

The algorithm for the lower bound-oriented calculation for parameter A for a given information word length l and a desired minimum Hamming distance of b is depicted in Fig. 2. In general, the algorithm structure equals the brute force approach, whereby each component is optimized as presented above. In both approaches we iterate over the odd values for A (line 16). In contrast to the brute force approach, the algorithm needs no nested loop for the iteration of code words, but only one per value for A (line 4). The two optimizations in (12) and (13) are applied in lines 6 and 2.

V. Results

We compared the lower bound-oriented approach[3] with the brute force for verification. Table I contains a qualitative comparison of the calculated values for A per l and $d_{\min}$. Each row contains the results of the brute force approach (upper line) and the results of the lower bound calculation

[3]Our code is available: https://brics-db.github.io/ANCodeParameter/

TABLE I
SMALLEST A FOR GIVEN l AND $d_{\min}(\mathbb{C}^A_{[0,2^l)})$ IN THE FORM A/k.

	$d_{\min}$							
l	2	3	4	5	6	7	8	9
4	3/2	19/5	89/7	557/10	4397/13	17779/15	119897/17	577881/20
	✓	✓	✓	*565/10* ✗	*4815/13* ✗	*23153/15* ✗	*169585/18* ✗	*1141127/21* ✗
8	3/2	29/5	185/8	1939/11	12779/14	55831/16	711805/20	5816601/23
	✓	✓	✓	✓	✓	✓	*835017/20* ✗	*7651641/23* ✗
12	3/2	37/6	267/9	3349/12	27825/15	214229/18	2686553/22	25781083/25
	✓	✓	✓	✓	✓	✓	✓	✓
16	3/2	47/6	393/9	4547/13	58659/16	687927/20	-	-
	✓	✓	✓	✓	✓	✓	*7933071/23*	*91221601/27*
20	3/2	53/6	555/10	6311/13	-	-	-	-
	✓	✓	✓	✓	*97569/17*	*1033069/20*	*15977383/24*	-
24	3/2	61/6	555/10	13837/14	-	-	-	-
	✓	✓	✓	✓	*157605/18*	*2053185/21*	*29533329/25*	-
28	3/2	71/7	737/10	17619/15	-	-	-	-
	✓	✓	✓	✓	*180111/18*	*3199675/22*	-	-
32	3/2	79/7	737/10	18613/15	-	-	-	-
	✓	✓	✓	✓	-	-	-	-

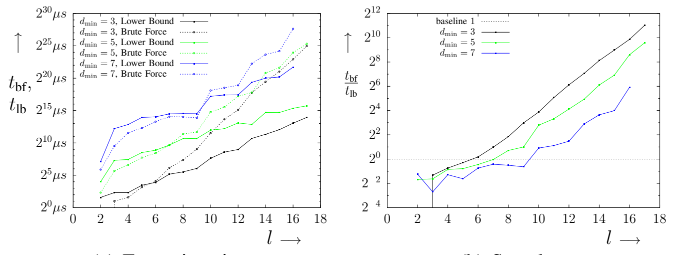
(lower line). A checkmark (✓) symbolizes a match between the brute force and the lower bound approach, missing values are indicated by a minus $(-)$[4]. As we can see, the lower bound approach computes the same values for A in most of the cases. Sometimes the lower bound calculation results in higher values, in particular for low information word lengths l. The reason for that is the lower bound consideration, but the approach never results in lower values for A.

Furthermore, we also compared the runtimes as highlighted in Fig. 3 on a regular CPU without vectorization. Both algorithms are implemented in C/C++ and compiled with gcc, whereby the execution was done single-threaded. As we can see in Fig. 3a, the brute force approach results in a less execution time than the lower bound approach for small values for l, whereas the differences are marginal. The reason is that the lower bound approach operates similarly to a brute force approach with small overhead in this case. However, for longer information word lengths l, the lower bound approach is much faster and the speedup of the lower bound approach grows exponentially compared to the brute force approach as shown in Fig. 3b. Only for gray values in Table I, the brute force was faster than the lower bound approach.

VI. CONCLUSION

In this paper, we considered AN coding with the challenge of computing an appropriate value for the parameter A, where the value depends on the information word length l and on the number of detectable bit flips b. Up-to-now, this calculation was done using an expensive brute force approach. To overcome that, we proposed a novel lower bound-oriented algorithm, which outperforms the brute force approach as clearly evaluated. With our approach, we are now able to efficiently compute the values for A for many combinations of l and b, which is e.g., necessary for the efficient application of AN coding for database systems as proposed in [12].

[4]At the time of publication, the calculations were not finished yet. In particular, for the brute force approach. An updated table can be found on our website: https://brics-db.github.io/ANCodeParameter/



(a) Execution time (b) Speedup

Fig. 3. Execution times for the calculation of A with given l, $d_{\min}$ with brute force (t_{bf}) and the lower bound approach (t_{lb}).

ACKNOWLEDGMENTS

This work was partly supported by the German Federal Ministry of Education and Research (BMBF) within the EXPLOIDS project under grant 16KIS0522K and by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden (cfaed)" (Resilience Path).

REFERENCES

[1] S. Y. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.
[2] Y. Kim, R. Daly, J. Kim, C. Fallin, J. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Int. Conf. on Software Architecture (ICSA)*, 2014, pp. 361–372.
[3] O. Mutlu, "The rowhammer problem and other issues we may face as memory becomes denser," in *Design, Automation and Test in Europe (DATE)*, 2017, pp. 1116–1121.
[4] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. R. Nassif, M. Shafique, M. B. Tahoori, and N. Wehn, "Reliable on-chip systems in the nano-era: lessons learnt and future trends," in *Des. Automation Conf. (DAC)*, 2013, pp. 99:1–99:10.
[5] S. Rehman, M. Shafique, and J. Henkel, *Reliable Software for Unreliable Hardware - A Cross Layer Perspective*. Springer, 2016.
[6] M. Shafique *et al.*, "Multi-layer software reliability for unreliable hardware," *it - Inf. Technol.*, vol. 57, no. 3, pp. 170–180, 2015.
[7] A. Avizienis, "Arithmetic error codes: Cost and effectiveness studies for application in digital system design," *IEEE Trans. Computers*, vol. 20, no. 11, pp. 1322–1331, 1971.
[8] M. Hoffmann, P. Ulbrich, C. Dietrich, H. Schirmeier, D. Lohmann, and W. Schröder-Preikschat, "A practitioner's guide to software-based soft-error mitigation using AN-codes," in *IEEE 15th Int. Symp. on High-Assurance Syst. Eng. (HASE)*, 2014, pp. 33–40.
[9] D. Kuvaiskii and C. Fetzer, "Δ-encoding: Practical encoded processing," in *DSN*, 2015, pp. 13–24.
[10] N. A. Rink and J. Castrillón, "Trading fault tolerance for performance in AN encoding," in *Proceedings of the Computing Frontiers Conf. (CF)*, 2017, pp. 183–190.
[11] U. Schiffel, "Hardware error detection using AN-codes," Ph.D. dissertation, Dresden University of Technology, 2011.
[12] T. Kolditz, D. Habich, W. Lehner, M. Werner, and S. T. J. de Bruijn, "AHEAD: adaptable data hardening for on-the-fly hardware error detection during database query processing," in *Proceedings of the 2018 Int. Conf. on Management of Data (SIGMOD)*, 2018, pp. 1619–1634.
[13] U. Güntzer and M. Paul, "Jump interpolation search trees and symmetric binary numbers," *Inf. Process. Lett.*, vol. 26, no. 4, pp. 193 – 204, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020019087900056
[14] C. Heuberger and H. Prodinger, "The hamming weight of the non-adjacent-form under various input statistics," *Periodica Mathematica Hungarica*, vol. 55, no. 1, pp. 81–96, 2007. [Online]. Available: https://doi.org/10.1007/s10998-007-3081-z
[15] OEIS Foundation Inc. (2017) The On-Line Encyclopedia of Integer Sequences. [Online]. Available: http://oeis.org